Assignment 4

Contributor List:

> Design: Yitong WANG, Yida TAO
>
> JUnit: Yitong WANG, Dingyuan XUE
>
> Document: Yida TAO
>
> Solution & Test: Siyi WANG, Sicen LIU

Question 1. BigBinary [Medium, 40 + 15 Points]

> Contributors: Yitong WANG, Siyi WANG

In the course CS202: Computer Organization, it is introduced that computers store data in the binary form.

YeeTone is a student of CS major but not good at solving binary number arithmetic problems, so he asks you for help. Fortunately, he is very familiar with Java programming. If you can help YeeTone solve problems by designing a class in Java, he will be very happy!

Please design a Java class named BigBinary to give a hand to YeeTone. Notice that in order to let YeeTone really trust you have the ability to help him solve the problems, YeeTone does not want you to use some classes in the libraries of Java such as `java.math.BigInteger` in this problem.

Constructor

```
public BigBinary(int[] bits, boolean positive);
```

Design a constructor with two parameters:

- An int array `bits` consisting of 0s and 1s, meaning the corresponding bit values. `0 <= bits.length <= 1,5000`. Notice that in order to ensure the data security, you should not copy the reference but the values of `bits`.
- A boolean `positive`, indicating whether the sign of BigBinary object is positive.

In the following examples, `^` means power in math instead of XOR. For instance, 3^5 = 3*3*3*3*3=243.

Example 1

```
bits = {0, 1, 0, 1, 1, 0}, positive = false
```

The bigBinary can express the binary number `-10110`, whose decimal number is `-(2^4 + 2^2 + 2^1) = -22`.

Example 2

```
bits = {}, positive = true
```

The bigBinary can express the binary number `0`, whose decimal number is `0`.

Example 3

```
bits = {1, 0, 0, 0, 1, 1}, positive = true
```

The bigBinary can express the binary number `100011`, whose decimal number is `2^5 + 2^1 + 2^0 = 35`.

Methods

- `public String toString();`

This method can return the BigBinary in a binary format. Notice that if there exists at least one `1` in the binary form, then the leading 0s should not be output; otherwise return `"0"`. For example, if the BigBinary is `-0011011`, the `toString()` method should return `"-11011"`; if the BigBinary is `-00000`, the `toString()` method should return `"0"`.

Sample code:

```
public static void main(String[] args) {
    int[] b1 = {0, 0, 1, 0, 1};
    int[] b2 = {1, 1, 0};
    int[] b3 = {};
    int[] b4 = {1, 1, 1, 0, 0};
    int[] b5 = {0, 1};

    BigBinary bigBinary1 = new BigBinary(b1, true);
    BigBinary bigBinary2 = new BigBinary(b2, false);
    BigBinary bigBinary3 = new BigBinary(b3, false);
    BigBinary bigBinary4 = new BigBinary(b4, true);
    BigBinary bigBinary5 = new BigBinary(b5, false);

    System.out.println(bigBinary1);
    System.out.println(bigBinary2);
    System.out.println(bigBinary3);
    System.out.println(bigBinary4);
    System.out.println(bigBinary5);
}
```

Sample Output:

```
101
-110
0
11100
-1
```

- `public BigBinary add(BigBinary bigbinary);`

After executing the method, the original BigBinary and the return value should be the sum of this BigBinary and the parameter `bigbinary`(i.e. this BigBinary + `bigBinary`).

- `public BigBinary minus(BigBinary bigbinary);`

After executing the method, the original BigBinary and the return value should be the difference between this BigBinary and the parameter `bigbinary`(i.e. this BigBinary - `bigBinary`).

- `public BigBinary multiply(BigBinary bigbinary);`

(Bonus) After executing the method, the original BigBinary and the return value should be the product of this BigBinary and the parameter `bigbinary`(i.e. this BigBinary * `bigBinary`).

- `public static BigBinary add(BigBinary b1, BigBinary b2);`

After executing the method, the return value should be the sum of BigBinary `b1` and BigBinary `b2`(i.e. `b1` + `b2`). BigBinary `b1` and `b2` should not be changed.

- `public static BigBinary minus(BigBinary b1, BigBinary b2);`

After executing the method, the return value should be the difference between BigBinary `b1` and BigBinary `b2`(i.e. `b1` - `b2`). BigBinary `b1` and `b2` should not be changed.

- `public static BigBinary multiply(BigBinary b1, BigBinary b2);`

(Bonus) After executing the method, the return value should be the product of BigBinary `b1` and BigBinary `b2`(i.e. `b1` * `b2`). BigBinary `b1` and `b2` should not be changed.

Hint for Bonus: Considering the data scale, the simple implementation may cause **TLE(Time Limit Exceeded)** and cannot pass the testcases. Please think about how to improve the efficiency of multiplication!

Question 2. Online Shopping [Medium, 60 + 5 Points]

> Contributors: Yida TAO, Sicen LIU, Dingyuan XUE

The advance in E-commerce has made online-shopping a pervasive and convenient daily experience. Customers could browse different online stores, each of which provides a list of products for customers to buy.
In this question, you'll build three classes, `Product`, `Store`, and `Customer`, to implement the online shopping process. See below for detailed requirements. Note that to implement the required functionality, you are free to add variables and methods that are not mentioned in this document; however, please **DO NOT** change the definitions of the existing variables and methods that we have specified in this document.

2.1 Product

Attributes

```
private static int cnt; // initialized to 0, and will increase by 1 when the constructor is called.
private int id;  // unique for each product and the value is set to cnt.
private String name;
private float price;
private ArrayList<Integer> ratings; // ratings from different customers; default is empty.
```

Constructor

```
public Product(String name, float price);
```

The constructor of the `Product` class.

- The `id` of the first product is 1. The given `price` is always valid.

Methods

```
public boolean setRating(int rating);
public float getAvgRating();
public String toString();
```

`public boolean setRating(int rating);`

Set `rating` to the product. The `rating` would be added to this product's rating list `ratings`.

- A `rating` should be within the range [1,5]; in other words, there are only 5 possible values for `rating`. If `rating` is not in this range, do not add it into `rating` and return `false`.

`public float getAvgRating();`

Return the average rating of this product, which is computed as the average of all the ratings it has received so far.

`public String toString();`

Return a string description of this product, in the format of `Product ID id, name, RMB price, Rating average-rating`, e.g., `Product ID 12345, Laptop, RMB 10000.00, Rating 4.5`.

- Round `price` to 2 decimal places and `rating` to 1 decimal place.

2.2 Store

Attributes

```
private static int cnt; // initialized to 0, and will increase by 1 when the
constructor is called.
private int id;  // unique for each store and the value is set to cnt.
private String name;
private ArrayList<Product> productList;
private float income;
```

Constructors

```
public Store(String name);
public Store(String name, ArrayList<Product> productList, float income);
```

There are two constructors of the `Store` class. One for constructing a new store, with `income = 0` and nothing in the `productList`. The other constructs an existing store with given `income` and `productList`.

- The `id` of the first store is 1. The given `income` and `productList`are always valid.

Methods

```
public boolean hasProduct(Product product);
public boolean addProduct(Product product);
public boolean removeProduct(Product product);
public ArrayList<Product> getProductList();
public void transact(Product product, int method);
```

`public boolean hasProduct(Product product);`

A method to determine whether this store has the given `product` . Return `true` if the `product` is in the `productList` of the store; otherwise, return `false`.

`public boolean addProduct(Product product);`

Add `product` to the `productList`; return a boolean indicating whether the operation succeeds.

- Suppose each `product`, which is uniquely identified by its `id`, could only appear once in the `productList` of a particular store. That is, the same `product` will not appear in the `productList` of multiple stores (this kind of invalid case will not happen); and in the same `productList`, a product appears not more than once.
- If a `product` already exists in `productList`, return `false` and `productList` remains the same; otherwise, add `product` to `productList` and return `true`.

`public boolean removeProduct(Product product);`

Remove `product` from `productList`; return a boolean indicating whether the operation succeeds.

- If `product` exists in `productList`, remove it from `productList` and return `true`; otherwise, return `false` and `productList` remains the same.

`public ArrayList<Product> getProductList();`

Return `productList`.

`public void transact(Product product, int method);`

This is an interface method for stores to handle customers' purchases or refunds. Suppose that all the arguments are valid here.

- `method = 0` means purchasing the `product` from this store. The `product` should be removed from the `productList` and the `income` of this store should increase by an amount equal to the `price` of the `product`.
- (Bonus) `method = 1` means refunding the `product` to the store. The `productList` and `income` of the store should also be updated accordingly (suppose that the store adds this product back to its `productList` and could re-sell this product).

2.3 Customer

Attributes

```
private static int cnt; // initialized to 0, and will increase by 1 when the
constructor is called.
private int id;  // unique for each customer and the value is set to cnt.
private String name;
private ArrayList<Product> shoppingCart; // The list of purchased products;
default is empty.
private float wallet;
```

Constructor

```
public Customer(String name, float wallet);
```

The constructor of the `Customer` class.

- The `id` of the first customer is 1. The given `wallet` is always valid.

Methods

```
public boolean rateProduct(Product product, int rating);
public void updateWallet(float amount);
public boolean purchaseProduct(Store store, Product product);
public void viewShoppingCart(SortBy sortMethod);
public boolean refundProduct(Product product);
```

`public boolean rateProduct(Product product, int rating);`

A customer can rate a `product` using this method.

- For invalid `rating`, return `false`; otherwise return `true`.

`public void updateWallet(float amount);`

Update the `wallet` of this customer. The `amount` could be positive (gaining money) or negative (consuming money). Assume that arguments are always valid.

`public boolean purchaseProduct(Store store, Product product);`

Purchase `product` from `store`.

- Return `true` if the `store` has this `product` and the customer has enough money in the `wallet` to purchase this `product`; return `false` otherwise. Note that the `shoppingCart` of this customer as well as his/her `wallet` should be updated accordingly.

`public void viewShoppingCart(SortBy sortMethod);`

Display the purchased products in the `shoppingCart` of this customer. The order of displaying is specified by `sortMethod`. Below we provide an Enum `SortBy`, which says that sorting could be performed by the `PurchaseTime`, `Rating`, or the `Price` of products.

```
public enum SortBy {
    PurchaseTime, Rating, Price
}
```

Suppose a customer Alice has purchased a few products from different stores.

```
Customer alice = new Customer("Alice", 20000);
// code for creating stores and products are ommitted
alice.purchaseProduct(store1, product_laptop);
alice.purchaseProduct(store1, product_table);
alice.purchaseProduct(store2, product_mouse);
alice.purchaseProduct(store3, product_phone);
```

Then, calling

```
alice.viewShoppingCart(SortBy.PurchaseTime);
```

will display (see `Product.toString()` for the format)

```
Product ID 2, Laptop, RMB 10000.00, Rating 4.5
Product ID 4, Table, RMB 300.00, Rating 4.3
Product ID 3, Mouse, RMB 100.00, Rating 3.0
Product ID 1, Phone, RMB 7000.00, Rating 4.5
```

Calling

```
alice.viewShoppingCart(SortBy.Rating);
```

will display

```
Product ID 3, Mouse, RMB 100.00, Rating 3.0
Product ID 4, Table, RMB 300.00, Rating 4.3
Product ID 2, Laptop, RMB 10000.00, Rating 4.5
Product ID 1, Phone, RMB 7000.00, Rating 4.5
```

- If the average rating is different without rounding but the same after rounding to one decimal place, then sort by the actual average rating (without rounding).
- If products have exactly the same average rating, they should be sorted by the purchase time.

Calling

```
alice.viewShoppingCart(SortBy.Price);
```

will display

```
Product ID 3, Mouse, RMB 100.00, Rating 3.0
Product ID 4, Table, RMB 300.00, Rating 4.3
Product ID 1, Phone, RMB 7000.00, Rating 4.5
Product ID 2, Laptop, RMB 10000.00, Rating 4.5
```

- If products have the same price, they should be sorted by the purchase time.

`public boolean refundProduct(Product product);`

(Bonus): Return the `product` to the store where it was sold and get the money back. Return `true` if this customer has indeed purchased this product before and `false` otherwise. Note that the `shoppingCart` and `wallet` of this customer should be updated accordingly. In addition, the corresponding store should enable the refund process to update the `productList` and `income`.

Tips

- We prepared a bilingual helping document(双语帮助文档) for this assignment. Please refer to this document first if you have any trouble solving the problems.
- We prepared sample test cases for Q1 and Q2, along with a documentation explaining how you could use JUnit to run these tests locally. Although OJ uses a different set of test cases, the sample tests are useful for testing and debugging purposes.

Submission

You need to submit `BigBinary.java` for question 1, `Product.java`, `Store.java`, `Customer.java` and `SortBy.java` for question 2. **Notice that you should NOT contain any Chinese character in the submitted java files.**